

# **Leisure-Management-Web-App**

Jakob Pirker, 1231394

11. Juni 2016

# Inhaltsverzeichnis

<b>1</b>	<b>Motivation und Ziele</b>	<b>3</b>
1.1	Webservice . . . . .	3
1.2	Anpassbarkeit und Erweiterbarkeit . . . . .	3
1.3	Anforderungsübersicht . . . . .	3
<b>2</b>	<b>Problemstellungen und Lösungsansätze</b>	<b>4</b>
2.1	REST . . . . .	4
2.2	Überblick . . . . .	4
2.3	Backend: Spring . . . . .	5
2.3.1	Datenbank Anbindung . . . . .	5
2.3.2	Datenübertragung . . . . .	5
2.4	Frontend: AngularJS/Javascript . . . . .	5
<b>3</b>	<b>Backend</b>	<b>6</b>
3.1	Spring . . . . .	6
3.2	Architektur . . . . .	6
3.2.1	Controller . . . . .	6
3.2.2	Services . . . . .	6
3.2.3	Repositories . . . . .	6
3.3	Datenbankanbindung . . . . .	6
3.4	Frontend-Interface . . . . .	6
<b>4</b>	<b>Frontend</b>	<b>7</b>
4.1	Angular JS . . . . .	7
4.1.1	Zwei-Weg Datenbindung . . . . .	7
4.2	Tabs . . . . .	7
4.3	Tabellen . . . . .	7
4.4	Eingabeformen . . . . .	7
<b>5</b>	<b>Installation und Anmerkungen</b>	<b>8</b>
<b>6</b>	<b>Quellen</b>	<b>9</b>
6.1	AngularJS . . . . .	9
6.2	Abbildungen . . . . .	9

# 1 Motivation und Ziele

In fast allen Vereinen und Gemeinschaften werden öfters Freizeiten angeboten. Damit sind Veranstaltungen gemeint, bei denen Mitglieder gemeinsam (meistens für mehrere Tage) an einen bestimmten Ort fahren, um dort gemeinsam Zeit zu verbringen, um einer bestimmten Tätigkeit intensiv nachzukommen (z.B. Trainingslager) oder Ähnliches. Mit der Anzahl der Teilnehmer steigt auch der Organisationsaufwand, und erreicht oft ein Maß bei dem die Organisation durch herkömmliche Methoden wie Absprache und „Papier und Stift“ unwirtschaftlich bis unmöglich wird.

Ziel dieser Arbeit ist es, eine Basis zu schaffen, die die Organisation einer solchen Freizeit vereinfacht. Dies soll dadurch geschehen, dass die im Rahmen des Projekts implementierte Anwendung eine Struktur für die Organisation vorgibt, in der oft benötigte Elemente bereits integriert sind, und direkt für die Organisation verwendet werden können. Einige Beispiele hierfür wären: Auflistung aller Teilnehmer, Überprüfung der Anwesenheiten und Veranstaltungsbeiträge jedes Teilnehmers, Aufgabenverteilungen... Die grundlegenden Anforderungen sind in den nächsten Punkten aufgelistet.

## 1.1 Webservice

Eine wichtige Anforderung die sich aus der Verteilung von Aufgaben ergibt ist die Bedienbarkeit von einer beliebigen Stelle aus. Aus dieser Anforderung heraus hat sich die Ausführung der Anwendung als Web-Applikation ergeben. Daraus ergeben sich folgende Vorteile:

- Bedienbarkeit von jedem internetfähigen Endgerät mit Browser
- keine Notwendigkeit spezieller Software
- zentrale Datenverwaltung (Client-Server)

## 1.2 Anpassbarkeit und Erweiterbarkeit

Die vorgegebene Struktur soll kein absoluter Maßstab sein, sondern der Anwender soll (wo möglich) selbst entscheiden können, welche Elemente er verwendet, und welche nicht. Außerdem soll die Applikation von einer anderen Person (mit Informatik- Hintergrundwissen) möglichst einfach gewartet und an eine spezielle Freizeit angepasst werden können. Dies bezieht sich sowohl auf das Hinzufügen und Entfernen von Content auf der Client-Seite, als auch auf das Hinzufügen und Entfernen von zusätzlichen bzw. unnötigen Informations-Attributen und Funktionen auf der Server-Seite. Außerdem soll es möglich sein größere Informationselemente (z.B. Events) ohne viel Aufwand in die Datenstruktur zu integrieren.

## 1.3 Anforderungsübersicht

Es soll möglich sein Personen einzutragen, die Teilnehmer und/oder Mitarbeiter sein können, und für Personen Adressen festzulegen. Teilnehmer können einer Unterkunft

zugewiesen werden, und für diese kann wiederum eine Adresse festgelegt werden. Jedem Mitarbeiter können mehrere Aufgaben zugewiesen werden. Jeder Person können mehrere Zahlungen zugewiesen werden, mehrere Zahlungen können wiederum einem Zahlungsdepot zugewiesen werden.

## 2 Problemstellungen und Lösungsansätze

### 2.1 REST

*Representational State Transfer (abgekürzt REST, seltener auch ReST) bezeichnet ein Programmierparadigma für verteilte Systeme, insbesondere für Webservices.<sup>1</sup>*

REST verweist auf einige Prinzipien, die vorausgesetzt werden, damit der implementierte Service als RESTful bezeichnet werden kann. Bei den Prinzipien handelt es sich um: Client-Server, Zustandslosigkeit, Caching, eine Einheitliche Schnittstelle, Mehrschichtige Systeme und Code on Demand (optional). Im Zuge des Projekts wurde Wert darauf gelegt, dass das System diese Eigenschaften erfüllt.

### 2.2 Überblick

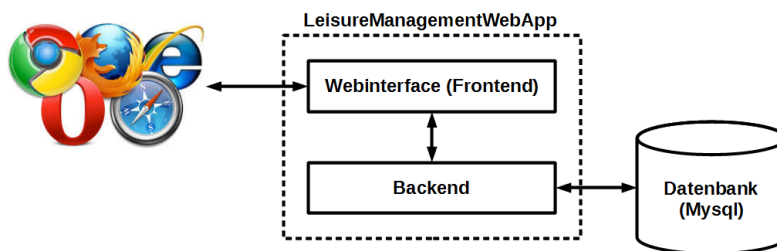


Abbildung 1: Grober Design Überblick

Wie die meisten Web-Applikationen besteht das Projekt aus einem Front- und einem Backend. Das Frontend (Client-Side) besteht aus einer Homepage, die von einem Browser aus aufgerufen wird. Es stellt eine Oberfläche zur Verfügung, mit deren Hilfe der Benutzer auf die Funktionalitäten der Applikation zugreifen kann. Das Frontend enthält jedoch nur Anzeige- und Transaktionslogik. Das bedeutet, dass es zwar Zugriff auf Funktionalitäten verschafft, und die Anzeigeelemente entsprechend der Benutzerinteraktion verändert, dann aber nur mittels eines http-Requests die entsprechenden Daten vom Backend anfordert und den zugehörigen Anzeigeelementen zuordnet. Die eigentliche Logik liegt im Backend (Server-Side). Es empfängt die Requests des Frontends, und sendet Daten als Antwort auf die Requests. So gut wie alle für die Organisation der Freizeit notwendigen Daten werden in einer Datenbank abgelegt. Das Backend fungiert auch als Schnittstelle zur Datenbank.

<sup>1</sup>[https://de.wikipedia.org/wiki/Representational\\_State\\_Transfer](https://de.wikipedia.org/wiki/Representational_State_Transfer)

## 2.3 Backend: Spring

Das Backend wurde in Java implementiert. Hier wurde Spring<sup>2</sup> als unterstützendes Framework gewählt, da es einen sehr großen Funktionsumfang hat. Die beiden für dieses Projekt wichtigsten Features werden im Folgenden kurz beschrieben.

### 2.3.1 Datenbank Anbindung

Spring stellt mittels Spring-Data einige sehr praktische Werkzeuge zur Verfügung mit denen auf relationale Datenbanken zugegriffen werden kann. Es werden JPA-Schnittstellen zur Verfügung gestellt, die die Zuordnung und Übertragung von Objekten zu Datenbankeinträgen sehr stark vereinfachen.

### 2.3.2 Datenübertragung

Spring unterstützt außerdem einfache Wege, mittels denen URL's einfach auf Methoden gemappt werden können. Die empfangenen http-Requests können durch Lesen der URL-Parameter oder der Daten im http-Body in nutzbare Daten umgewandelt werden.

## 2.4 Frontend: AngularJS/Javascript

Zur Erstellung des Frontends wurde das JavaScript Framework AngularJS verwendet. AngularJS erweitert HTML um Attribute. Diese Attribute schaffen erstens eine sehr gute Schnittstelle zwischen den HTML-Elementen und der dahinter liegenden JavaScript Logik und zweitens ermöglichen sie eine dynamische Anpassung der HTML-Struktur des Basisdokuments.

---

<sup>2</sup><https://spring.io>

## 3 Backend

### 3.1 Spring

#### 3.1.1 Dependency Injection

Wie bereits erwähnt wurde als Framework für das Java-Backend Spring verwendet. Ein sehr nützliches Design-Konzept von Spring ist die Dependency Injection.

*Als Dependency Injection wird in der objektorientierten Programmierung ein Entwurfsmuster bezeichnet, welches die Abhängigkeiten eines Objekts zur Laufzeit reglementiert: Benötigt ein Objekt beispielsweise bei seiner Initialisierung ein anderes Objekt, ist diese Abhängigkeit an einem zentralen Ort hinterlegt – es wird also nicht vom initialisierten Objekt selbst erzeugt.<sup>3</sup>*

Die Abhängigkeit von einem anderen, existierenden Objekt wird im Code meistens über die `@Autowired`-Annotation definiert. Durch dieses Entwurfsmuster ist es möglich, kurzen, gut leserlichen und gut wartbaren Code zu schreiben. Einerseits weil dadurch sehr viele Dinge nicht implementiert werden müssen, andererseits weil die Komponenten der Applikation klar getrennt werden können, da die Verknüpfungen automatisch zur Laufzeit erstellt werden.

#### 3.1.2 JavaBeans

### 3.2 Architektur

#### 3.2.1 Controller

#### 3.2.2 Services

#### 3.2.3 Repositories

### 3.3 Datenbankbindung

### 3.4 Frontend-Interface

---

<sup>3</sup>[https://de.wikipedia.org/wiki/Dependency\\_Injection](https://de.wikipedia.org/wiki/Dependency_Injection)

## **4 Frontend**

### **4.1 Angular JS**

#### **4.1.1 Zwei-Weg Datenbindung**

### **4.2 Tabs**

### **4.3 Tabellen**

### **4.4 Eingabeformen**

## **5 Installation und Anmerkungen**



## 6 Quellen

- REST: [https://de.wikipedia.org/wiki/Representational\\_State\\_Transfer](https://de.wikipedia.org/wiki/Representational_State_Transfer)
- REST: <https://spring.io/understanding/REST>
- JPA: [https://de.wikipedia.org/wiki/Java\\_Persistence\\_API](https://de.wikipedia.org/wiki/Java_Persistence_API)

### 6.1 Spring

- <https://spring.io>
- Dependency Injection: [https://de.wikipedia.org/wiki/Dependency\\_Injection](https://de.wikipedia.org/wiki/Dependency_Injection)

### 6.2 AngularJS

- <https://de.wikipedia.org/wiki/AngularJS>
- <https://angularjs.org/>
- <http://www.w3schools.com/angular/>

### 6.3 Abbildungen

- Abbildung 1 (Browser Zusammenstellung): <http://vestavialibrary.org/wp-content/uploads/2016/02/web-browsers.jpg>